CLAIMS

We claim:

1. A method comprising:

receiving an entered string; and

determining how likely a word was to have been entered as the string based on at least one edit operation that converts a first character sequence of arbitrary length in the word to a second character sequence of arbitrary length in the string.

2. A method as recited in claim 1, wherein the first character sequence has a first length and the second character sequence has a second length that is different than the first length.

3. A method as recited in claim 1, wherein the first character sequence has multiple characters and the second character sequence has multiple characters.

4. A method as recited in claim 1, wherein the first character sequence has a first number of multiple characters and the second character sequence has a second number of multiple characters that is different from the first number of multiple characters.

5. A method as recited in claim 1 and further comprising determining how likely the word is to have been generated.

6. A method as recited in claim 1 and further comprising conditioning the edit operation on a position that the edit occurs at within the word.

7. A method as recited in claim 1 and further comprising identifying the string as potentially incorrect.

8. A method as recited in claim 1 and further comprising correcting the string to the word.

9. A computer readable medium having computer-executable instructions that, when executed on a processor, perform the method as recited in claim 1.

10. A method comprising:

receiving an entered string *s*; and

determining a probability P(*s*|*w*) expressing how likely a word *w* was to have been incorrectly entered as the string *s* based on one or more edit operations that convert first arbitrary-length character sequences α_1, α_2, α_3, ..., α_n in the word *w* to corresponding second arbitrary-length character sequences β_1, β_2, β_3, ..., β_n in the string *s*, wherein:

$$P(s|w) = P(\beta_1|\alpha_1) * P(\beta_2|\alpha_2) * P(\beta_3|\alpha_3) * \ldots * P(\beta_n|\alpha_n)$$

11. A method as recited in claim 10, wherein lengths of corresponding first and second character sequences are different.

12. A method as recited in claim 10 and further comprising determining how likely the word *w* is to have been generated.

13. A method as recited in claim 10 and further comprising conditioning the edit operations on positions that the edits occur at within the word.

14. A method as recited in claim 10 and further comprising correcting the string *s* to the word *w*.

15. A method as recited in claim 10 and further comprising identifying the string *s* as potentially incorrect.

16. A computer readable medium having computer-executable instructions that, when executed on a processor, perform the method as recited in claim 10.

17. A method comprising:

receiving an entered string *s*; and

determining a probability P(*s*|*w*) expressing how likely a word *w* was to have been incorrectly entered as the string *s,* by partitioning the word w and the string s and computing probabilities for various partitionings, as follows:

$$P(s \mid w) = \sum_{R \in Part(w)} P(R \mid w) \sum_{\substack{T \in Part(s) \\ |T|=|R|}} \prod_{i=1}^{|R|} P(T_i \mid R_i)$$

where Part(w) is a set of possible ways of partitioning the word w, Part(s) is a set of possible ways of partitioning the string s, R is a particular partition of the word w, and T is a particular partition of the string s.

18. A method as recited in claim 17 and further comprising selecting the partition that returns a highest probability.

19. A method as recited in claim 17 and further comprising determining how likely the word w is to have been generated.

20. A method as recited in claim 17 and further comprising correcting the string s to the word w.

21. A method as recited in claim 17 and further comprising identifying the string s as potentially incorrect.

22. A computer readable medium having computer-executable instructions that, when executed on a processor, perform the method as recited in claim 17.

23. A method comprising:

receiving an entered string *s*; and

determining a probability P(*s*|*w*) expressing how likely a word *w* was to have been incorrectly entered as the string *s*, by partitioning the word w and the string s and computing probabilities for various partitionings, as follows:

$$P(s|w) = \max_{R \in Part(w),\ T \in Part(s)} P(R|w) * \prod_{i=1}^{|R|} P(T_i|R_i)$$

where Part(*w*) is a set of possible ways of partitioning the word *w*, Part(*s*) is a set of possible ways of partitioning the string *s*, R is a particular partition of the word *w*, and T is a particular partition of the string *s*.

24. A method as recited in claim 23 and further comprising omitting the term P(R|w) from the computation of P(*s*|*w*).

25. A method as recited in claim 23 and further comprising setting terms $P(T_i|R_i) = 1$ whenever $T_i = R_i$.

26. A method as recited in claim 23 and further comprising determining how likely the word *w* is to have been generated.

27. A method as recited in claim 23 and further comprising correcting the string *s* to the word *w*.

28. A method as recited in claim 23 and further comprising identifying the string *s* as potentially incorrect.

29. A computer readable medium having computer-executable instructions that, when executed on a processor, perform the method as recited in claim 23.

30. A method comprising:

receiving an entered string *s*; and

determining a probability P(*s*|*w*) expressing how likely a word *w* was to have been incorrectly entered as the string *s*, by partitioning the word w and the string s and finding a partition R of the word w and a partition T of the string s such that $\prod_{i=1}^{|R|} P(T_i \mid R_i)$ is maximized.

31. A method as recited in claim 30 and further comprising determining how likely the word *w* is to have been generated.

32. A method as recited in claim 30 and further comprising correcting the string *s* to the word *w*.

33. A method as recited in claim 30 and further comprising identifying the string *s* as potentially incorrect.

34. A computer readable medium having computer-executable instructions that, when executed on a processor, perform the method as recited in claim 30.

35. A method for training an error model used in a spell checker, comprising:

determining, given a <wrong, right> training pair and multiple single character edits that convert characters in one of the right or wrong strings to characters in the other of the right or wrong strings at differing costs, an alignment of the wrong string and the right string that results is a least cost to convert the characters;

collapsing any contiguous non-match edits into one or more common error regions, each error region containing one or more characters that can be converted to one or more other characters using a substitution edit; and

computing a probability for each substitution edit.

36. A method as recited in claim 35, wherein the assigning comprises assessing a cost of 0 to all match edits and a cost of 1 to all non-match edits.

37. A method as recited in claim 35, wherein the single character edits comprises insertion, deletion, and substitution.

38. A method as recited in claim 35, further comprising collecting multiple <wrong, right> training pairs from online resources.

39. A method as recited in claim 35, further comprising expanding each of the error regions to capture at least one character on at least one side of the error region.

40. A program embodied on a computer readable medium, which when executed, directs a computer to perform the following:

receive an entered string; and

determine how likely an expected string was to have been entered as the entered string based on at least one edit operation that converts a first character sequence of arbitrary length in the expected string to a second character sequence of arbitrary length in the entered string.

41. A program as recited in claim 40, wherein the first character sequence has a first length and the second character sequence has a second length that is different than the first length.

42. A program as recited in claim 40, wherein the first character sequence has multiple characters and the second character sequence has multiple characters.

43. A program as recited in claim 40, wherein the first character sequence has a first number of multiple characters and the second character sequence has a second number of multiple characters that is different from the first number of multiple characters.

44. A program as recited in claim 40, further comprising computer-executable instructions that directs a computer to determine how likely the expected string is to have been generated.

45. A program as recited in claim 40, further comprising computer-executable instructions that directs a computer to perform, depending upon how likely an expected string was to be incorrectly entered as the entered string, one of the following: (1) leave the entered string unchanged, (2) autocorrect the entered string into the expected string, or (3) offer a list of possible corrections.

46. A spell checker program, embodied on a computer-readable medium, comprising the program of claim 40.

47. A language conversion program, embodied on a computer-readable medium, comprising the program of claim 40.

48. A word processing program, embodied on a computer-readable medium, comprising the program of claim 40.

49. A program embodied on a computer readable medium, which when executed, directs a computer to perform the following:

(1) receive an entered string *s*;

(2) for multiple words *w* in a dictionary, determine:

(a) how likely a word *w* in a dictionary is to have been generated, P(*w*|*context*); and

(b) how likely the word *w* was to have been entered as the string *s*, P(*s*|*w*), based on at least one edit operation that converts a first character sequence of arbitrary length in the word to a second character sequence of arbitrary length in the string; and

(3) maximize P(*s*|*w*)*P(*w*|*context*) to identify which of the words is most likely the word intended when the string s was entered.

50. A program as recited in claim 49, wherein the determination (2) is performed for all words in the dictionary.

51. A program as recited in claim 49, further comprising computer-executable instructions that directs a computer to either (1) leave the string unchanged, (2) autocorrect the string into the word, or (3) offer a list of possible corrections.

52. A spell checker program, embodied on a computer-readable medium, comprising the program of claim 49.

53. A language conversion program, embodied on a computer-readable medium, comprising the program of claim 49.

54. A spell checker comprising:

a source model component to determine how likely a word *w* in a dictionary is to have been generated; and

an error model component to determine how likely the word *w* was to have been incorrectly entered as the string *s* based on arbitrary length string-to-string transformations.

55. A spell checker as recited in claim 54, wherein the string-to-string transformations involve conversion of a first character sequence of a first length into a second character sequence of a second length that is different than the first length.

56. A spell checker as recited in claim 54, wherein the string-to-string transformations involve conversion of a first character sequence with multiple characters into a second character sequence with multiple characters.

57. A spell checker as recited in claim 54, wherein the string-to-string transformations involve conversion of a first character sequence having a first number of multiple characters into a second character sequence having a second number of multiple characters that is different from the first number of multiple characters.